The Design of Network Services for Advanced Collaborative Environments

Han Gao[•] Rick L. Stevens[•◊] Michael E. Papka [•◊]
[•]*Department of Computer Science*
University of Chicago
[◊]*Mathematics and Computer Science Division*
Argonne National Laboratory
{hangao, stevens, papka}@cs.uchicago.edu

Abstract

Advanced collaborative environments are one of the most important tools for interacting with colleagues distributed around the world. However, heterogeneous characteristics such as network transfer rates, computational abilities, and hierarchical systems make the seamless integration of distributed resources a challenge. This paper proposes the design of two network services, Collaborative Environment Network Service Architecture (CENSA) and Infrastructure (CENSI), that embed network services into various systems intelligently and elastically and support seamless advanced collaborative environments. We present a multilayered model for their current utilization and future development. We describe various network services and discuss some open issues.

1. Introduction

Advances in collaboration [1, 2] and Grid technology [3, 4] have motivated development of advanced collaborative environments. Many individuals and groups are now using such environments as part of their everyday collaborations with colleagues distributed around the world. However, an even larger number of individuals and groups are hampered from participating fully in collaborative environments because of restrictions such as insufficient network bandwidth, legacy systems technology, heterogeneous network systems, and hardware devices.

Consider the following scenario. Alice wishes to join a group meeting via the Access Grid while she is traveling. However, her PDA can receive and send text information only. One solution is a network service that transforms audio signal to text information and vice versa, so that she can interact with her colleagues freely. This scenario (Figure 1) illustrates that content transformation is one of most important features of network service for advanced collaborative environments such as the Access Grid.

Consider another scenario. An NSF grant is shared by five universities distributed across the country. Some of the larger universities have multiple individuals working as part of the team, while the smaller universities have individual PIs. The team members have agreed that regular discussions are much more productive than yearly meetings and have decided to adopt the Access Grid as a platform for such discussions. During the first meeting, the wide-area network connection of one of the smaller universities is unable to handle the amount of network traffic required to sustain an Access Grid session. This scenario demonstrates another issue we are addressing in this paper: network transfer rates.

The solution we propose is introducing one or more sets of network services into a collaborative environment, which can bypass the traffic links and route a new efficient stream topology.

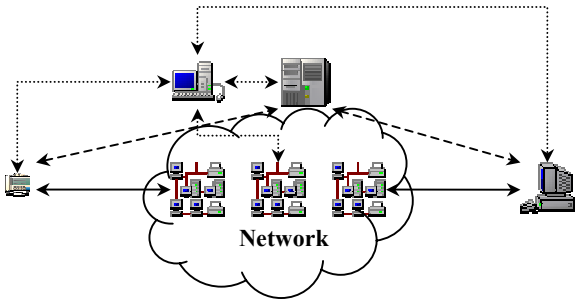

Figure 1: Content transformation scenario: the solid arrow lines denote the view of end users; the segment tracks denote actual data stream route; the dotted lines denote the management view of the whole framework.

2. Related Work

One recent work in this area is Web services, through which means software components are loosely coupled, encapsulate distinct functionality, and are distributed and accessible by using standard Internet protocols. Basically, Web services use UDDI (Universal Discovery, Description and Integration) [5], WSDL (Web Services Description Language) [6], SOAP [7], and XMLRPC [8] for the description, discovery, and messaging protocols of Web services. This infrastructure is still in its infancy. Furthermore, the technology is driven by business needs or business-oriented processes, which in some cases differ greatly from our own.

Grid technologies such as the Globus Toolkit® [9, 10] and Open Grid Services Architecture (OGSA) [11] define an extensible set of services—Grid services [12, 13] – that can be aggregated in various ways by virtual

organizations [14]. The semantics defines standard mechanisms for creating, naming, and discovering transient Grid service instances. It also provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities. This infrastructure is focused mainly on scientific and technical computing and the transition from research to business purposes.

CANS [15] is an application-level infrastructure for injecting application-specific components into the network. It integrates individual software components efficiently and dynamically, self-adapts components in responses of nonlocal changes and distributed system conditions, and supports legacy applications and services. However, this infrastructure needs a global management for a specific collaborative environment.

Bramly et al. [16] implement a DOE Common Component Architecture on top of a Globus Toolkit Grid framework. They use XML as an extensible and powerful tool to describe software components and running instances of these components. This allows description and user interfaces to be generated from the specification dynamically. The service components can be replaced or extended easily. Bramley and his colleagues also build distributed applications using a graphical "drag-and-drop" interface, Web-based interface, scripting language such as Python, or existing tools such as Matlab.

3. Collaborative Environment Network Service Architecture

An advanced collaborative environment is composed of a set of service nodes, links, and end users. For seamless integration, network services should have the following properties:

- Service-oriented: network services include computational and storage resources, networks, databases, software objects and hardware devices.
- Soft-state: because of the heterogeneous characteristics of systems, services can find a reasonable architecture to fit in. Each component will exist and be maintained by a mixture of explicit and probabilistic means.
- Decentralized [17]: services are managed by the specific advanced collaborative environment with local management modules.
- Reusable [18]: according to the various characteristics and sophisticated conditions, each application can adjust the architecture to meet the specific requirements. Each layer and component can be plugged in and out easily.
- Self-adaptable[15]: the network service should provide a global adaptation during the running. It should reconfigure the necessary components to handle dynamic and nonlocal changes.
- Transparent: the running procedure of network service is as transparent as possible to each end user in the collaborative environment. After the user application invokes the network service, steps are processed by the other modules in our framework.
- Stream-capability: based on the capabilities of today's networks and processors the framework should have the ability to deal with large scale data streams for advanced collaborative environments.

According to this view and the examples we discussed above, we propose a general architecture for development and utilization of network services, which we call CENSA (see Figure 2). Our theoretical model is a four-layered, hierarchical system structure:

1. Application layer: colleagues can plug their various resources, applications, and services from different advanced collaborative domains into this layer.
2. Content interceptor layer: this layer translates applications, resources, and services to description files defined by specific schemas. It also forwards the content information to the stream layer.
3. Stream routing layer: this layer records information from a data stream to a suitable data structure from the content layer and obtains link information from the lower layer.
4. Network link layer: this layer provides registered network physical links and reflectors to the upper layer.

The architecture also includes a management module, which provides several functionalities, such as discovering, registering, and matching. Two types of API are also incorporated: one is used for communication between adjacent layers, and the other is used for interacting with the management module.

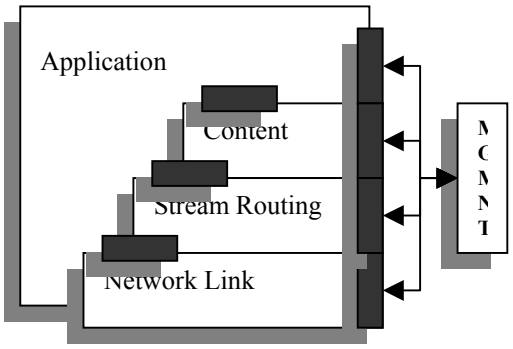

Figure 2: Collaborative Environment Network Service Architecture (CENSA): dark grey blocks represent APIs; arrow lines connect each layer to each corresponding management module.

This architecture meets different requirements and situations for advanced collaborative environments. For example, we have several approaches for solving the problem presented by our second scenario. One solution is that, when the content layer detects the current data stream causing the network traffic, it forwards this information to the application layer to see whether the upper layer can reproduce a lower stream to reduce the network traffic. Another solution is to have the content layer invoke a transcoding network service to translate the higher bandwidth stream to a lower one. A third solution is to have the stream routing layer reconfigure the stream routing topology via some hardware devices such as reflectors, routers, and switches registered in the network links layer.

4. Collaborative Environment Network Service Infrastructure

Advanced collaborative environments have attracted increasing attention from research communities, with concomitant higher demands and requirements. We present one ideal model for a collaborative environment network infrastructure, which we call CENSI (see Figure 3).

CENSI defines a standard infrastructure for network services, Grid services, Web services, and others. It supports the semantics of network services and ensures that they run smoothly under this environment. Both CENSA and CENSI construct a complete system for network services.

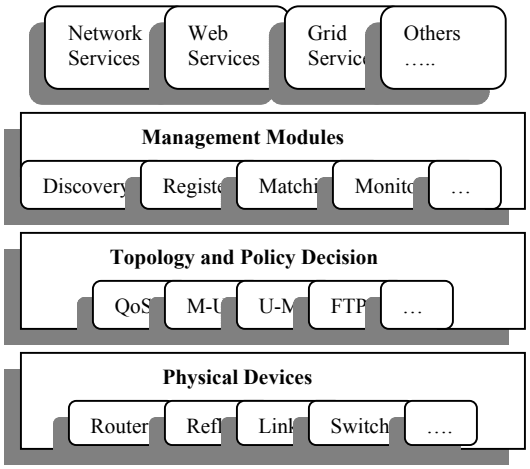

Figure 3: Collaborative Environment Network Service Infrastructure (CENSI).

5. Example: Capability Negotiation

To illustrate the overall operation and procedure of the CENSA and CENSI, we return to our second scenario. The problem is that because of technical limitations, some end users cannot join the Access Grid session. To enable all the colleagues to collaborate, we introduce the following services:

1. Capabilities Initialization – Before starting the Access Grid session, users provide a set of capability description files that represent the current environment. This process may involve loading a previously created configuration; using a discovery tool to automatically probe the local machine and networking environment; using an editor to manually configure the capabilities; or some combination of all of these. When the process is complete, the XML document representing the current capability description is loaded into the Access Grid Network Service interface tool.
2. Venue Logon – This process involves a security authentication between the AG client and the Virtual Venue server.
3. Network Services Invocation – The AG Venue returns a session bundle, which comprises the user's current capabilities and the offerings as presented by the Venue. If necessary, an AG network service handle is included in this session bundle, and it will be passed to matchmaker for capability negotiation.
4. Matchmaking – The matchmaker examines the session bundle, iterating through the capabilities and offerings, searching for conflicts.
5. Resolution – The final phase of capability/offering conflict resolution uses a database of available network services and resource descriptions for the available instances of those services, along with a set of decision rules, to determine what network services are available to resolve the conflicts in this session.

AG Client Reconfiguration – The clients receive the solution, which resolves the conflict with the new offering. The client software running in the user's environment is reconfigured, if necessary, to comply with the new offerings and the user is able to participate in the collaboration.

This example illustrates the whole running procedure and how to select network service. In the following section, we use the Access Grid as a reference model to discuss some useful network services.

6. Network Services

New network services are created frequently, so our proposed framework must be flexible, deployable, and user-extensible. The framework is based on two services: matchmaker and stream topology. We will integrate the network services into the Access Grid as a reference model. This will hasten the acceptance and use of the network services framework, to the benefit of all colleagues such as AG users. Moreover, it will directly address the problems introduced in our two scenarios. We list services under development or being considered:

- Network bridging – Many institutions are enabling multicast to support the Access Grid. While considerable progress has been made on the issues regarding reliable deployment of wide-area native multicast routing, it is still not always possible to deploy this technology as quickly or seamlessly as desired. The bridging service provides a mechanism to allow access AG applications to use a standard multicast service model whether or not the underlying network supports native IP multicast, using a variety of bridging or routing technologies.
- Audio transcoding – It provides the conversion service between different resolutions of audio streams.
- Multicast beacon – The multicast beacon service provides multicast connectivity information. This service can be used to detect multicast failures, which can then multicast network performance that can be used to generate heuristic-based measurements of network performance.
- Network audio fallback to phone – When network audio fails, AG meetings need to continue uninterrupted. For this purpose, audio teleconferencing is provided by standard analog phone lines. This service detects network failures and configures and dials phone calls to provide a smooth transition from network to phone audio.
- Video subsampling – It subsamples the input video stream to a smaller size. The subsampling parameters must be specified.
- Video stream compositing – It takes multiple video streams and composites them into one video stream. This can be used to conserve bandwidth, to associate steams by source, or in a production setting to reduce the number of streams. The generated stream specifications must be defined. Also ,the compositing method must be specified.
- Audio stream mixing –Multiple audio streams are mixed together to provide a single audio stream. The generated stream specifications must be specified.
- Network audio monitor – It provides analytical information on audio streams. This information can be used by other network services to modify the audio streams for different desired effects.
- Network audio equalization – It performs equalization based on input from the network audio monitor service to correct audio streams from sites that don't conform to a specified set of parameters.
- Closed captioning – It provides a mechanism for audio to be translated into text and broadcast in real time with the video streams.
- Language translation – It translates audio and/or text from one language to another.

The first two issues are discussed in the following sections: matchmaking and stream topology.

7. General Matchmaker

Collaboration needs to be an open effort among individuals, unlimited by the technology that brings them together. Network services are designed to facilitate collaboration. One open issue, however, is how to determine which network service is best for some specific collaborative environment.

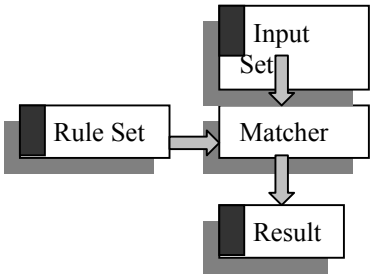

Figure 4: General matchmaker: each dark grey block represent a schema for each capability description files and rule set files.

In Figure 2, each layer connects its specific management module. As one important component in this management module, matchmaker is required for each layer in network service selection. (see Figure 4).

7.1. Schemas Design

For efficiency and portability, we need define schemas of each description file, data file, and criterion. For the input set, schemas define a set of independent parameters, which can describe one object clearly and nonredundantly. The schema of the rule set defines the priority and comparison functions. These functions depend on different properties and attributes of various matchings. Some languages [19] have been developed for object description.

7.2. Matching Algorithms

We propose a heuristic algorithm based on linear algebra for resource matching. A resolution vector of a resource object is represented as $\vec{v} = (v_1, v_2, ..., v_n) \in R^n$ where $v_1, v_2, ..., v_n$ are n independent parameters and R^n is an n-dimensional vector space. For each capability set, we can compose at least one segment—a set of vectors in this space. Suppose two vectors $\vec{v}_1, \vec{v}_2 \in R^n$ need to match with each other via network service. The network service can be represented as an $n \times n$ transformation matrix T related with $\vec{v}_1$ and $\vec{v}_2$:

$$\vec{v}_1 = T\vec{v}_2$$

Suppose we have two sets of resolutions: $S_1^n = \{\vec{v}_{1,1}, \vec{v}_{1,2}, ..., \vec{v}_{1,i}, ..., \vec{v}_{1,n}\}, S_2^n = \{\vec{v}_{2,1}, \vec{v}_{2,2}, ..., \vec{v}_{2,j}, ..., \vec{v}_{2,m}\}$, $S_1^n, S_2^n \subseteq R^n$. Our goal is obtaining the best common resolution for each set and, if network services are needed, the corresponding transformation matrix:

$$(\vec{v}^*, T^*) = \begin{cases} (\arg\max\{\vec{v}_i | \vec{v}_i \in S^*\}, I), S^* \neq \Phi \\ (\arg\max\{\vec{v}_i | \vec{v}_i \in S^{**}\}, T_{12}), S^* = \Phi, S^{**} \neq \Phi \end{cases}$$

where S^* is a common resolution set, S^{**} is the common resolution set after transformation, and

$$T_{1,2} = T(S_1^n, S_2^n) = T_{i,j}\Big|_{\arg\max\{\vec{v} | \vec{v} \in S^{**}\}}, \quad \text{where } \vec{v} = \vec{v}_{1,i} = T_{i,j}\vec{v}_{2,j}$$

Consider multiple sets situation. Suppose we have k sets of capabilities, $S_1^n, S_2^n, ..., S_k^n, S_i^n \subseteq R^n, i = 1, ..., k.$ We obtain independent capability sets first:

$$\vec{S}_1^n \perp \vec{S}_2^n \perp ... \perp \vec{S}_l^n, \quad l \leq k$$

Our goal is the best common solution vector and the transformation matrices

$$(\vec{v}^*, T^*) = \begin{cases} (\arg\max\{\vec{v}_i | \vec{v}_i \in S^*\}, \Phi), S^* \neq \Phi \\ (\arg\max\{\vec{v}_i | \vec{v}_i \in S^{**}\}, T^*), S^* = \Phi, S^{**} \neq \Phi \end{cases}$$

$$T^* = \{T_{1,2}, T_{1,3}, ..., T_{i,j}, ..., T_{l-1,l}\}, T_{i,j} = T(\vec{S}_i^n, \vec{S}_j^n), i, j = 1, ..., l$$

Also, we need to define the function *argmax* details under specific conditions.

8. Streaming Topology

Network traffic such as limited bandwidth is a key issue for advanced collaborative environments. Network services are introduced to bypass or remedy this weak link. However, where and how to put the specific network services into the existing collaboration is still a challenge. Currently, we have two topologies, one based on the user's view and the other on the physical link. The upper topology is composed of end users, applications, and services (Figure 5) while the lower topology is composed of routers, switches, and other transferring and linking devices. Our goal is to find a mapping between these two topologies and to determine the lowest cost – the definition of which depends on different situations. The basic idea is that for an active collaboration environment, we borrow non active services to make a seamless integration.

Figure 5: Snapshot of Access Grid active status on June 6, 2003, yellow dots represent sites running the NLANR multicast beacon.

We also describe this network service mathematically. Suppose we have two sets of nodes: active session V and nonactive session V'. For the active session, we have a graph $G(V, E)$. If it is not connected, we need to borrow a node or set of nodes from V' and compose a new connected graph $G''(V'', E'')$. We denote the cost of graph as $C(G)$. The optimal solution is a minimum-cost graph:

$$C_{opt} = \min_{V'', E''}(C(G''(V'', E'')))$$

9. Conclusion

This paper proposes a design for advanced collaborative environments in such a way as to enable the addition of arbitrary new network services. We summarize the properties of network services and present an infrastructure, CENSA and CENSI, based on the matchmaker and stream topology modules. Using the Access Grid as a reference model, we give some general ideas for utilization and development of such an advanced collaborative environment tool, which supports a seamless integration of heterogeneous services.

Acknowledgments

We thank the entire Futures Laboratory: Justin Binns, Terry Disz, Ed Frank, Mary Fritsch, Mark Hereld, Randy Hudson, Dave Jones, Ivan Judson, Susanne Lefvert, Ti Leggett, Eric Olson, Robert Olson, and Thomas Uram. Thanks to Joe Insley for producing the image of Access Grid sites. Funding for this work has been provided by the National Science Foundation NMI program, with additional funding for the Access Grid from the U.S. Department of Energy under Contract W-31-109-Eng-38, and Microsoft Research.

References

[1] L. Childers, T. L. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens, "ActiveSpaces on the Grid: The Construction of Advanced Visualization and Interaction Environments," in *Parallelldatorcentrum Kungl Tekniska Högskolan Seventh Annual Conference (Simulation and Visualization on the Grid)*, vol. 13, *Lecture Notes in Computational Science and Engineering*, B. Engquist, L. Johnsson, M. Hammill, and F. Short, Eds. Stockholm, Sweden: Springer-Verlag, 1999, pp. 64-80.

[2] T. L. Disz, R. Evard, M. W. Henderson, W. Nickless, R. Olson, M. E. Papka, and R. Stevens, "Designing the Future of Collaborative Science: Argonne's Futures Laboratory," *IEEE Parallel and Distributed Technology Systems and Applications*, vol. 3, pp. 14 -21, 1995.

[3] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1999.

[4] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Physics Today*, vol. 55, pp. 42-47, 2002.

[5] "Universal description discovery and integration (UDDI)." 2001.

[6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," 2001.

[7] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," 2000.

[8] A. Brown, M. Fuchs, J. Robie, and P. Wadler, "XML Schema: Formal Description," 2001.

[9] I. Foster and C. Kesselman, "The Globus Project: A Status Report," in *IPPS/SPDP '98 Heterogeneous Computing Workshop*, 1998, pp. 4-18.

[10] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *Intl J. Supercomputer Applications*, vol. 11, pp. 115-128, 1997.

[11] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002.

[12] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kessellman, "Grid Service Specification," Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002.

[13] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *IEEE Computer*, vol. 35, pp. 37-46, 2002.

[14] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal Supercomputer Applications*, vol. 15, 2001.

[15] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, Adaptive Network Service Infrastructure," in *USENIX Sysmposium on Internet Technologies and System (USITS)*. San Francisco, CA, 2001.

[16] R. Bramley, K. Chiu, S. Diwan, D. Gannon, M. Govindaraju, N. Mukhi, B. Temko, and M. Yechuri, "A Component Based Services Achitecture for Building Distributed Applications," presented at Ninth IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, PA, 2000.

[17] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemmi, "Decentralizing Control and Intelligence in Network Management," in *Fourth International Symposium on Integrated Network Management*, A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, Eds. Santa Barbara, CA: Chapman & Hall, 1995, pp. 4-16.

[18] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, "Distributed Intelligent Agents," *IEEE Expert*, vol. 11, pp. 36-46, 1996.

[19] R. Raman, M. Livny, and M. H. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," in *High Performance Distributed Computing*. Chicago, IL: IEEE Computer Society, 1998, pp. 140 - 146.